LISTING OF THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:

assigning a sequence of time intervals to each software package of the plurality of software packages, the sequence of time intervals assigned to a particular software package of the plurality of software packages not overlapping the sequence of time intervals assigned to any other software package of the plurality of software packages;

executing a subset of the plurality of software packages, each respective software package in the subset plurality of software packages being executed during predetermined time intervals defined by the sequence of time intervals assigned to the respective software package in the subset of the plurality of software packages.

2. (Previously Presented) The method of claim 1 further comprising the step of utilizing one or more tests to identify the plurality software packages that are valid, and wherein the subset of the plurality of software packages includes only valid software packages.

3. (Previously Presented) The method of claim 2 wherein a given test of the one or more tests for validity is a one's complement checksum test of a software package's program memory data.

4. (Previously Presented) The method of claim 2 wherein a given software package of the plurality of software packages is assigned a dedicated memory region, a given test of the one or more tests for validity being whether an address returned for an initialization procedure of the

given software package of the plurality of software packages lies within the dedicated memory region of the given software package of the plurality of software packages.

5. (Currently Amended) The method of claim 4 wherein the given test of the one or more tests is whether the address is returned for the initialization procedure ~~of the~~ of the given software package of the plurality of software packages within a predetermined time.

6. (Previously Presented) The method of claim 2 wherein a given software package of the plurality of software packages is assigned a dedicated memory region, the dedicated memory region of the given software package of the plurality of software packages including a stack memory region and/or a heap memory region, a given test of the one or more tests for validity being whether the stack memory region and/or the heap memory region assigned during the execution of an initialization procedure of the given software package of the plurality of software packages and various associated entry points lies within the dedicated memory region assigned to the given software package of the plurality of software packages.

7. (Previously Presented) The method of claim 6 wherein the given test of the one or more tests is whether the stack memory region and/or the heap memory region and the various associated entry points are returned within a predetermined time.

8. (Previously Presented) The method of claim 1 wherein a given software package of the plurality of software packages is assigned a dedicated memory region.

9. (Previously Presented) The method of claim 8 wherein the dedicated memory region assigned to the given software package of the plurality of software packages includes a stack memory region in which a stack of the given software package of the plurality of software packages resides.

10. (Previously Presented) The method of claim 1 wherein a given software package of the plurality of software packages includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

11. (Previously Presented) The method of claim 10 wherein a given background task of the background tasks is an infinite loop.

12. (Previously Presented) The method of claim 10 wherein the given software package of the plurality of software packages causes power utilized in executing the given software package of the plurality of software packages to be minimized after completion of the background tasks.

13. (Previously Presented) The method of claim 1 wherein a failure in the execution of a given software package of the plurality of software packages causes information to be logged in a failure log.

14. (Previously Presented) The method of claim 13 wherein a failure in execution is linked to the given software package of the plurality of software packages that caused the failure.

15. (Previously Presented) The method of claim 13 wherein quality of performance in executing the given software package of the plurality of software packages is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in the failure log, the execution of the given software package of the plurality of software packages being subject to a plurality of execution options, an execution option being selected on the basis of the values of the one or more performance-quality parameters.

16. (Original) The method of claim 15 wherein the plurality of execution options are user configurable.

17. (Previously Presented) The method of claim 15 wherein the one or more performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in the failure log.

18. (Original) The method of claim 1 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

19. (Previously Presented) The method of claim 1 wherein each software package of the plurality of software packages is assigned a memory block, a given software package of the plurality of software packages being configured to read data only from zero or more memory blocks associated with other software packages of the plurality of software packages, the zero or more memory blocks readable by the given software package of the plurality of software packages being either predetermined or determined during execution of the given software package of the plurality of software packages in accordance with a set of one or more rules.

20. (Currently Amended) The method of claim 1 wherein each software package of the plurality of software packages is assigned a memory block, a given software package of the plurality of software packages being configured to write data only to zero or more memory blocks associated with other software packages of the plurality of software ~~packages~~ packages, the zero or more memory blocks writeable by the given software package of the plurality of software packages being either predetermined or determined during execution of the given software package of the plurality of software packages in accordance with a set of one or more rules.

21. (Currently Amended) The method of claim 1 wherein an executive software package enforces the discipline that each of the respective software packages in the subset ~~of the~~ of the plurality of software packages is executed only during the time intervals defined by the sequence of time intervals assigned to the respective software package in the subset of the plurality of software packages, the executive software package determining when the execution of any one of the respective software packages in the subset of the plurality of software packages extends into a time interval defined by the sequence of time intervals assigned to at least one different software package in the subset of the plurality of software packages and performs a remedial action.

22. (Previously Presented) The method of claim 1 wherein a presence of the subset of the plurality of software packages is detected.

23. (Previously Presented) The method of claim 1 wherein one or more software packages of the plurality of software packages are independently compiled, linked, and loaded.

24. (Previously Presented) The method of claim 1 wherein each software package of the plurality of software packages has a stack that is selected prior to executing the software package.

25. (Original) Apparatus for practicing the method of claim 1.

26. (Previously Presented) Apparatus for repetitively executing a plurality of software packages at a plurality of rates, the apparatus comprising:

a means for generating and assigning a sequence of time intervals to each software package of the plurality of software packages, the sequence of time intervals assigned to a particular software package of the plurality of software packages not overlapping the

sequence of time intervals assigned to any other software package of the plurality of software packages;

a means for executing a subset of the plurality of software packages, each respective software package in the subset of the plurality of software packages is executed during predetermined time intervals defined by the sequence of time intervals assigned to the respective software package in the subset of the plurality of software packages.

27. (Previously Presented) The apparatus of claim 26 further comprising a means for utilizing one or more tests to identify the software packages that are valid, and wherein the subset of the plurality of software packages includes only valid software packages.

28. (Previously Presented) The apparatus of claim 27 wherein a given test of the one or more tests for validity is a one's complement checksum test of a software package's program memory data.

29. (Previously Presented) The apparatus of claim 27 wherein a given software package of the plurality of software packages is assigned a dedicated memory region, a given test of the one or more tests for validity being whether an address returned for an initialization procedure of the given software package of the plurality of software packages lies within the dedicated memory region of the given software package of the plurality of software packages.

30. (Previously Presented) The apparatus of claim 29 wherein the given test of the one or more tests is whether the address is returned within a predetermined time.

31. (Previously Presented) The apparatus of claim 27 wherein a given software package of the plurality of software packages is assigned a dedicated memory region, the dedicated memory region of the given software package of the plurality of software packages including a stack memory region and/or a heap memory region, a given test of the one or more tests for validity

being whether the stack memory region and/or the heap memory region assigned during the execution of an initialization procedure of the given software package of the plurality of software packages and various associated entry points lies within the dedicated memory region of the given software package of the plurality of software packages.

32. (Previously Presented) The apparatus of claim 31 wherein the given test of the one or more tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.

33. (Previously Presented) The apparatus of claim 26 wherein a given software package of the plurality of software packages is assigned a dedicated memory region.

34. (Previously Presented) The apparatus of claim 33 wherein the dedicated memory region of the given software package of the plurality of software packages includes a stack memory region in which a stack of the given software package of the plurality of software packages resides.

35. (Previously Presented) The apparatus of claim 26 wherein a given software package of the plurality of software packages includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

36. (Previously Presented) The apparatus of claim 35 wherein a given background task of the background tasks is an infinite loop.

37. (Previously Presented) The apparatus of claim 35 wherein the given software package of the plurality of software packages causes power utilized in executing the given software package of the plurality of software packages to be minimized after completion of the background tasks.

38. (Previously Presented) The apparatus of claim 26 wherein a failure in the execution of a given software package of the plurality of software packages causes information to be logged in a failure log.

39. (Previously Presented) The apparatus of claim 38 wherein a failure in execution is linked to the given software package of the plurality of software packages that caused the failure.

40. (Previously Presented) The apparatus of claim 38 wherein quality of performance in executing the given software package of the plurality of software packages is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in the failure log, the execution of the given software package of the plurality of software packages being subject to a plurality of execution options, an execution option being selected on the basis of the values of the one or more performance-quality parameters.

41. (Original) The apparatus of claim 40 wherein the plurality of execution options are user configurable.

42. (Previously Presented) The apparatus of claim 40 wherein the one or more performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a the failure log.

43. (Original) The apparatus of claim 26 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

44. (Previously Presented) The apparatus of claim 26 wherein each software package of the plurality of software packages is assigned a memory block, a given software package of the

plurality of software packages being configured to read data only from zero or more memory blocks associated with other software packages of the plurality of software packages, the zero or more memory blocks readable by the given software package of the plurality of software packages being either predetermined or determined during execution of the given software package of the plurality of software packages in accordance with a set of one or more rules.

45. (Previously Presented) The apparatus of claim 26 wherein each software package of the plurality of software packages is assigned a memory block, a given software package of the plurality of software packages being configured to write data only to zero or more memory blocks associated with other software packages of the plurality of software packages, the zero or more memory blocks writeable by the given software package of the plurality of software packages being either predetermined or determined during execution of the given software package of the plurality of software packages in accordance with a set of one or more rules.

46. (Previously Presented) The apparatus of claim 26 wherein an executive software package enforces the discipline that each of the respective software packages in the subset of the plurality of software packages is executed only during time intervals defined by the sequence of time intervals assigned to the respective software package in the subset of the plurality of software packages, the executive software package determining when the execution of any one of the respective software packages in the subset of the plurality of software packages extends into a time interval defined by the sequence of time intervals assigned to at least one different software package in the subset of the plurality of software packages and performs a remedial action.

47. (Previously Presented) The apparatus of claim 26 wherein a presence of the subset of the plurality of software packages is detected.

48. (Previously Presented) The apparatus of claim 26 wherein one or more software packages of the plurality of software packages are independently compiled, linked, and loaded.

49. (Previously Presented) The apparatus of claim 26 wherein each software package of the plurality of software packages has a stack that is selected prior to executing the software package.